УДК 61:004.651(075.8)

РОЗРОБКА КЛІНІЧНОЇ ДІАГНОСТИЧНОЇ СИСТЕМИ, ЩО ҐРУНТУЄТЬСЯ НА ПРАВИЛАХ, ПОБУДОВАНИХ МЕТОДОМ ПОСЛІДОВНОГО ПОКРИТТЯ

О. О. Стаханська

ДВНЗ «Тернопільський державний медичний університет імені І. Я. Горбачевського МОЗ України»

У роботі розглянуто питання обчислювальної складності алгоритму індукції правил на основі алгоритму послідовного покриття при розробці клінічної діагностичної системи. Встановлені оцінки підтверджено експериментально як із зміною кількості атрибутів, так і обсягу наборів навчальних даних.

Ключові слова: диференційна діагностика, прийняття рішень, індукція правил.

РАЗРАБОТКА КЛИНИЧЕСКОЙ ДИАГНОСТИЧЕСКОЙ СИСТЕМЫ, ОСНОВАННОЙ НА ПРАВИЛАХ, ПОСТРОЕННЫХ МЕТОДОМ ПОСЛЕДОВАТЕЛЬНОГО ПОКРЫТИЯ

О. О. Стаханська

ГВУЗ «Тернопольський государственный медицинский университет имени И. Я. Горбачевского МЗ Украины»

В работе рассматриваются вопросы вычислительной сложности алгоритма индукции правил на основе алгоритма последовательного покрытия при разработке клинической диагностической системы. Установленные оценки подтверждены экспериментально как с изменением количества атрибутов, так и объема наборов учебных данных.

Ключевые слова: дифференциальная диагностика, принятие решений, индукция правил.

DEVELOPMENT OF CLINICAL DIAGNOSTIC SYSTEM BASED ON RULES WITH HELP OF METHOD OF SEQUENTIAL COVERING

O. O. Stakhanska

SHEI «Ternopil State Medical University by I. Ya. Horbachevsky of MPH of Ukraine»

The work deals with the computational complexity of the rule induction algorithm based on sequential covering when developing clinical diagnostic systems. Established evaluation confirmed experimentally as a change in the amount of attributes, and the volume of training data sets.

Key words: differential diagnosis, making decision, rules induction.

Сьогодні медичне діагностування може виконуватися автоматично з використанням комп'ютеризованих систем та алгоритмів. Такі системи переважно називають діагностичними системами підтримки прийняття рішень або медичними діагностичними системами. Вони належать до загального класу клінічних систем підтримки прийняття рішень [9–11]. Метою таких систем є системний супровід лікаря у процесі диференційної діагностики. Багато з таких систем можуть надавати результати навіть коли не вистачає даних, тобто в умовах невизначеності, і що найважливіше – вони не обмежені щодо кількості інформації, яку можуть зберігати та обробляти [3–8].

У даній роботі ми досліджуватимемо класифікатор, що ґрунтується на правилах, в якому модель знань представлено множиною правил IF-THEN.

Математичне означення класифікаційних правил. Традиційне означення IF-THEN-правила наведено в роботах [1, 5]. Математично задача індукції класифікаційних правил формулюється таким чином. Маємо множину D, що містить N наборів на-

вчальних даних. При цьому кожен *i*-й набір $(A_1^i, A_2^i, ..., A_p^i, C^i)$ складається з вхідних даних – атрибутів $A_1, ..., A_p$ та вихідних даних – атрибуту класу C. Можна припустити, що атрибути $A_1, ..., A_p$ приймають лише категоріальні значення. Атрибут класу C приймає одне з K дискретних значень: $C \in \{1, ..., K\}$. Метою є прогнозування класифікаційним правилом значення атрибуту класу C на основі значень атрибутів $A_1, ..., A_p$.

Класифікаційним правилом R називається імплікація вигляду: $R: \underset{j \in S}{\wedge}(A_j \text{ is } a_j^*) \Rightarrow C = c^*$. Тут $S \subseteq \{1, ..., p\}$ – деяка підмножина індексів атрибутів.

При цьому слід максимізувати точність прогнозування атрибуту класу, а саме $P\{C = c\}$ для довільного $c \in \{1, ..., K\}$. В результаті ми повинні отримати множину правил для кожного $c \in \{1, ..., K\}$ відповідно, що в антеседенті містять умови включення для категоріальних атрибутів, а в консеквенті значення $c \in \{1, ..., K\}$.

Метою роботи є дослідити питання обчислювальної складності методу індукції класифікаційних правил послідовним покриттям при розробці клінічної експертної системи.

Алгоритм послідовного покриття. Використаємо алгоритм послідовного покриття, описаний в роботі Han, 2001. Припускаємо, що усі атрибути – категоріальні.

Алгоритм послідовного покриття

Вхідні дані:

D – множина навчальних наборів даних $(A_1^i, A_2^i, ..., A_p^i, C^i)$

Att_vals – множина всіх атрибутів $A_1, ..., A_p$ та їх можливих значень $A_i \in \{a_i^1, a_i^2, ..., a_i^{K_i}\}$

Вихідні дані: *Rule_set* – множина класифікаційних правил.

Метод:

1. Множина класифікаційних правил *Rule_set* = {}
2. Для кожного класу *c*
3. Розпочати цикл "до"
4. Побудувати нове класифікаційне правило *Rule* = *Добути_одне_правило (D, Att_vals, c)*
5. Вилучити набори навчальних даних з *D*, що покриваються правилом *Rule*
6. Виконувати цикл з кроку 3 до настання *термінальної умови*
7. Додати нове правило до множини класифікаційних правил:

Rule_set = *Rule_set* + *Rule*

8. Кінець циклу з кроку 2
9. Множина навчальних правил в *Rule_set*

В основу методу *Добути_одне_правило (D, Att_vals, c)* покладена міра приросту інформації для побудови правил логіки першого порядку FOIL (First Order Inductive Learner). Метод є ітераційною процедурою по усіх атрибутах $A_1, ..., A_p$.

Припустимо, що ми вже маємо класифікаційне правило:

R: IF *condition* THEN *class* = *c*.

Метою кожного кроку $i = \overline{1, p}$ є кон'юнкція умови *condition* за рахунок умови *condition'* вигляду $(A_i = a_i^j)$. Тут $j \in \{1, ..., K_i\}$. Тобто нове правило матиме вигляд:

R': IF *condition* AND *condition'* THEN *class* = *c*.

Згідно з методом FOIL *condition'* вибирається з умови мінімізації міри:

$$FOIL\_Gain = pos' \times (\log_2 \frac{pos'}{pos' + neg'} - \log_2 \frac{pos}{pos + neg}) \quad (1)$$

Тут *pos(neg)* – число позитивних (негативних) навчальних наборів, що покриваються правилом *R*, *pos'(neg')* – число позитивних (негативних) навчальних наборів, що покриваються правилом *R'*. Під позитивними (негативними) навчальними наборами для певного правила маємо на увазі навчальні набори з умовою консеквенту, які задовольняють (не задовольняють) умови антеседенту правила.

Міра (1) сприяє побудові правил, що мають більшу точність і при цьому покривають якомога більше позитивних навчальних наборів.

Розрахунок обчислювальної складності алгоритму. З аналізу алгоритму послідовного покриття бачимо, що обчислювальна складність визначається добутком кількості можливих значень атрибуту класу *K* (кількість ітерацій зовнішнього циклу) та обчислювальної складності процедури *Добути_одне_правило (D, Att_vals, c)*, яку виконують всередині кожного циклу.

Процедура *Добути_одне_правило (D, Att_vals, c)* включає виконання *p* ітерацій. На кожній ітерації для певного атрибуту A_i проводять розрахунок міри $FOIL\_Gain$ для кожного з K_i значень атрибуту. Тобто внутрішнє тіло циклу в процедурі *Добути-*

_одне_правило (D, Att_vals, c) виконують $\sum_{i=1}^{p} K_i$ разів. Міру *FOIL _Gain* обчислюють в результаті 4-х SQL-запитів, складність яких можна оцінити величиною $O(\log(N))$ (див. документацію до MySQL 5.0 – http://dev.mysql.com/doc/refman/5.0/en/select-speed.html). Отже, в цілому процедура *Добути_одне_правило (D, Att_vals, c)* має обчислювальну складність $O\left(\sum_{i=1}^{p} K_i \times \log(N)\right)$.

Підсумовуючи, маємо обчислювальну складність всього алгоритму послідовного покриття порядку

$$O\left(K \times \sum_{i=1}^{p} K_i \times \log(N)\right). \qquad (1)$$

Чисельний експеримент. Для прикладу використано експериментальну базу даних біохімічних аналізів залежно від виду політравми. Навчальні набори містять 21 категоріальний атрибут та 6 різних значень атрибуту класу.

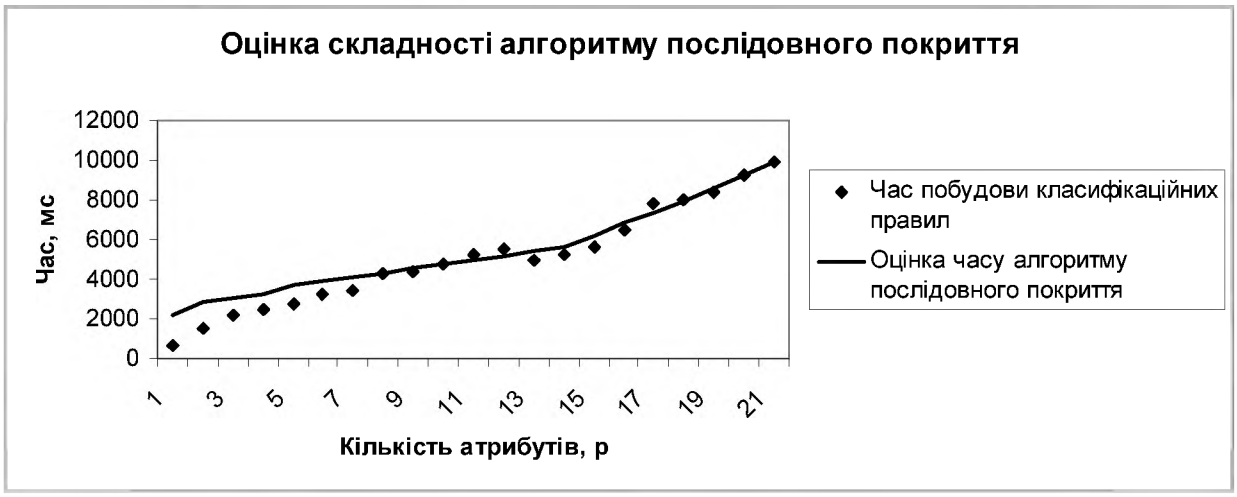
Дослідження проводили у двох напрямках: змінюючи число атрибутів в навчальних наборах; змінюючи кількість навчальних наборів. При цьому в обох випадках приходимо до оцінки часу побудови множини правил вигляду (1):

$$23. \times K \times \sum_{i=1}^{p} K_i \times \log(N) + 1550. \qquad (2)$$

У додатках наведено побудовані класифікаційні правила.

На рисунку 1 представлено результати чисельного експерименту.

а)



б)

Рис. 1. Порівняння результатів чисельного експерименту з оцінкою складності алгоритму (2).

Оптимізація побудови копій класифікаційних правил. Як зазначалося вище, з метою створення "глибоких" копій об'єктів у програмі використано технологію Java Object Serialization (JOS). Це – загальний підхід, який полягає в тому, що відбувається записування об'єкта до масиву з використанням ObjectOutputStream, а згодом створення копії об'єкта за допомогою ObjectInputStream. В результаті створюється повністю окремий об'єкт з повністю відмінними об'єктами, на які він посилається. Саме такий підхід первинно було використано в програмній реалізації алгоритму.

На жаль, тут виникають проблеми, а саме:

– даний метод працює лише, коли об'єкти, що копіюються, а також об'єкти, на які йдуть прямі або непрямі посилання, підтримують серіалізацію. Тобто вони реалізують інтерфейс java.io.Serializable. На щастя, досить лише декларації implements java.io.Serializable;

– технологія Java Object Serialization є повільною і її використання для створення глибокої копії вимагає як серіалізації, так і десеріалізації;

– реалізацію потоку байтового масиву, що входить до пакета java.io, розроблено для досить загального використання для даних різних розмірів і для забезпечення безпеки в багатопотокових середовищах. Ці характерні особливості однак уповільнюють ByteArrayOutputStream і меншою мірою ByteArrayInputStream.

З метою вирішення певних з перелічених проблем (особливо третьої) використовуємо підхід, запропонований у роботі [http://javatechniques.com/blog/faster-deep-copies-of-java-objects/] і який полягає в альтернативних реалізаціях класів ByteArrayOutputStream та ByteArrayInputStream, що робить три простих оптимізації:

– ByteArrayOutputStream за припущенням починається з 32-байтного масиву для виводу. Далі при запису контенту до потоку розмір масиву при потребі збільшується (або до затребуваного розміру, або розмір просто подвоюється). Отже, первинний розмір масиву в 32 байти означає, що створюється багато малих масивів, які потім копіюються і зрощуються при записі даних. Отже, є проста оптимізація – створити масив з більшим початковим розміром;

– усі методи класу ByteArrayOutputStream є синхронізованими. В цілому це правильно, але можна бути певними, що лише один потік має доступ до ByteArrayOutputStream. Вилучення синхронізації дасть певне пришвидшення. Методи класу ByteArrayInputStream залишаються й надалі синхронізованими;

– метод toByteArray() створює і повертає копію байтового масиву з потоку. Це проста ідея, яка полягає у тому, що в противному випадку, коли ми створюємо інший окремий байтовий масив для копіювання в нього, то йде сповільнення зарахунок виконання додаткової роботи.

Таким чином, даний альтернативний підхід було використано в методі обчислення міри private double FOIL_Gain(…), а саме: копія правила створюється викликом спеціально створеного методу:

rule_prime = (Rule)rule.copyOptimized().

Провівши чисельний експеримент (рис. 2), встановлено значення оцінки часу виконання алгоритму:

$$21.\times K\times\sum_{i=1}^{n}K_i\times log(N)+1550 \qquad (3)$$

Рис. 2. Порівняння результатів чисельного експерименту на основі оптимізованого копіювання правил з оцінкою складності алгоритму (3).

Отже, в цілому бачимо певну оптимізацію часу виконання алгоритму (рис. 3), яка відчутніше проявляється при збільшенні обсягу наборів навчальних даних.

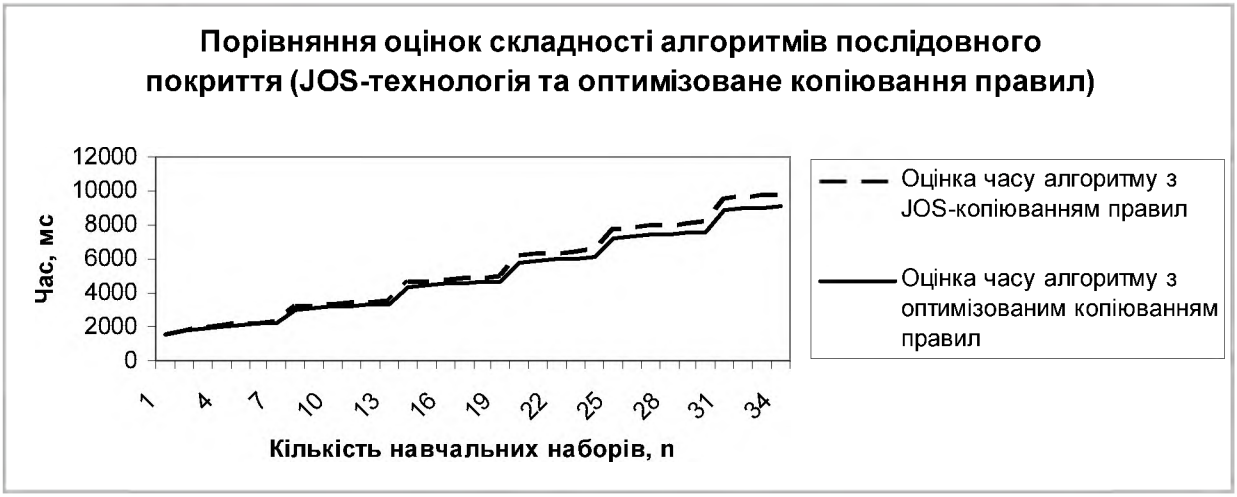


Рис. 3.

Висновки. У роботі розглянуто питання обчислювальної складності алгоритму послідовного покриття побудови класифікаційних правил в клінічній діагностичній системі. Визначено оцінку обчислювальної складності та її чисельно досліджено на прикладі клінічної експертної системи. При цьому запропоновано та підтверджено вплив оптимізації процедури копіювання правил, особливо при збільшенні обсягу наборів навчальних даних.

Література

1. Han J. Data Mining: Concepts and Techniques / J. Han, M. Kamber // Morgan Kaufmann, San Francisco, 1st edition, 2001.
2. Hastie T. The Elements of Statistical Learning / Hastie T., Tibshirani R., Friedman J. H. // Springer, New York, 1st edition, 2001.
3. Ordonez C. Comparing association rules and decision trees for disease prediction / Ordonez C. // ACM HIKM Workshop. – 2006. – P. 17–24.
4. Ordonez C. Integrating K-means clustering with a relational DBMS using SQL, IEEE Transactions on Knowledge and Data Engineering (TKDE) / C. Ordonez. – 2006. – Vol. 18(2). – P. 188–201.
5. Ordonez C. Models for association rules based on clustering and correlation / C. Ordonez // Intelligent Data Analysis. – 2009. – Vol. 13(2). – P. 337–358.
6. Quinlan J. R. Induction of decision trees / J. R. Quinlan // Machine Learning. – 1986. – Vol. 1. – P. 81–106.
7. Quinlan J. R. C4.5: Programs for Machine Learning / J. R. Quinlan // Morgan Kaufmann, 1993.
8. Classification and Regression Trees / L. Breiman, J. Friedman, R. Olshen, C. Stone // Wadsworth International Group, 1984.
9. Марценюк В. П. О программной среде проектирования интеллектуальных баз данных / В. П. Марценюк, Н. О. Кравец // Клиническая информатика и телемедицина. – 2004. – № 1. – С. 47–53.
10. Математичні моделі в системі підтримки прийняття рішень страхового забезпечення лікування онкологічних захворювань: підхід на основі динаміки Гомперца / В. П. Марценюк, І. Є. Андрущак, І. С. Гвоздецька, Н. Я. Климук // Доповіді Національної академії наук України. –2012. – № 10. – С. 34–39.
11. Марценюк В. П. О модели онкологического заболевания со временем пребывания на стадии в соответствии с распределением Гомперца / В. П. Марценюк, Н. Я. Климук // Проблемы управления и информатики. Международный научно-технический журнал. – 2012. – №6. – С. 137–143.